オープンソースソリューションに対する不完全デバッグ環境を考慮した確率微分方程式モデルに基づく信頼性評価法 (不確実・不確定環境下における数理的意思決定とその周辺)

AUTHOR(S):

山本, 友基; 田村, 慶信; 山田, 茂

数理解析研究所講究録
第1802巻 2012年 113-119

オープンソースソリューションに対する不完全デバッグ環境を考慮した確率微分方程式モデルに基づく信頼性評価法

山口大学大学院・理工学研究科 山本 友基（Yuki Yamamoto）†
山口大学大学院・理工学研究科 田村 慶信（Yoshinobu Tamura）†
†Graduate School of Science and Engineering, Yamaguchi University
鳥取大学大学院・工学研究科 山田 茂（Shigeru Yamada）‡
‡Graduate School of Engineering, Tottori University

1 まえがき

ネットワーク環境を利用して開発されるオープンソースソフトウェア（Open Source Software，以下OSSと略す）は，世界中の誰もが開発に参加でき，ソースコードが公開され，誰でも自由に改変可能なソフトウェアであることから，組込みシステムやサーバ用途として広く採用され，急激に普及が広まっている[1]．また，オープン規格やOSSを利用することによって，電子行政機関がプライバシーや個人の自由を保護するとともに，市民が電子政府と情報をやり取りできるようにするのに役立つことから，EU加盟国を中心に欧米においても政府関係機関がOSSを支持する動きが広がっている[2]．さらに，導入コスト削減や短納期，他社の製品戦略に対して優位に立つという観点からも企業ソフトウェアにおける重要な選択肢の1つとしてOSSが注目されている．

一方，OSSの利用に関しては，未だに多くの不安が残されている．まず第1に，システム導入後のサポートや品質上の問題といった利用者側の一般的な不安である．第2に，OSSは本当にビジネスになるのか，オープンソースのソフトウェアを事業化することによって自社製のソフトウェア製品までが市場を失うことにならないか，といった開発者側の不安である[2]．特にサポートや品質上の問題については，OSSの普及を妨げる大きな要因として考えられている．本論文では，こうしたオープンソースプロジェクトの下で開発されているOSSから構成されるオープンソースソリューションに対する信頼性評価法を提案する．

従来から，ソフトウェア製品の開発プロセスにおけるテスト進捗管理や出荷品質の把握のための信頼性評価を行うアプローチとして，ソフトウェア故障の発生現象を不確定事象として捉えて確率・統計論的に取り扱う方法がとられている．その1つが，ソフトウェア信頼度成長モデル（Software Reliability Growth Model，以下SRGMと略す）である[3]．

また，OSSに対する現在の研究動向としては，設計工程や開発手法，セキュリティを対象とした文献はいくつか提案されているが[4, 5]，動的解析に基づいた信頼性評価に関する研究はほとんど行われていないのが現状である．さらに，OSSの信頼性評価として傾向分析に基づく事例研究としての文献についても，いくつか提案されているが，これらはOSSのもつ特有の開発形態を考慮した信頼性評価手法を提案したものではない[6, 7]．

OSSの開発形態を考えた場合，機能の不具合やセキュリティ上の脆弱性を修復するような，クリティカルなフォールトがいくつか発見され緊急に修正を施す必要がある場合に実施されるバグフィックスや，旧版にいくつかの細かい機能が追加されたり，性能が若干向上した場合に実施されるマイナーバージョンアップ，さらには，OSS自体の機能が大きく変更されたり，大型の新機能の追加されるメジャーバージョンアップが常に繰り返されている．こうした修正作業はOSSが無意味なものとみなされるまで継続される．OSSの開発形態は上記のような性質をもつことから，従来のような同一組織内で開発され，単体で動作するソフトウェアシステムとは開発環境が大きく異なる．

現在，複数のオープンソースコンポーネントを結合したオープンソースソリューションの利用が拡大している．こうしたオープンソースソリューションでは，オープンソースコンポーネントの規模も比較的大きな場合が通常であり，システム全体としても大規模化する傾向がある．また，複数のOSSから構成されているため，品質の面で全般的に問題となることが多く，1つの主要コンポーネントが全体のオープンソースソリューションとしての機能

自体を崩壊させる危険性も含んでいる．オープンソースソリューションの一例として，OSS である Apache HTTP Server，Apache Tomcat，MySQL といったサーバソフトウェアを組み合わせ，その基盤上に Java 言語で開発するようなシステムが挙げられる．このようなシステムの場合，各オープンソースコンポーネントの相互運用性確保が重要となる．

本論文では，こうした大規模オープンソースソリューション開発におけるテスト工程を対象とした確率微分方程式モデルを構築する．具体的には，オープンソースソリューションのテスト工程におけるソフトウェアフォールト発見率が時間とともに変化し，テスト工程初期段階においてオープンソースコンポーネントの結合状態が常に不規則な状態であると考え，オープンソースソリューション全体の安定性を示す成長パラメータを導入し，確率微分方程式 [8] に基づいた SRGM を構築する．さらに，有用な信頼性評価尺度として不完全デバッグ確率を導出し，実際の OSS のソフトウェアフォールト発見数データに対する数値例および感度分析結果を示すことにより，大規模オープンソースソリューションの信頼性評価法について考察する．

2 オープンソースソリューションに対する確率微分方程式モデル

まず，時刻 $t=0$ でオープンソースソリューションのテスト工程が開始され，任意の時刻 t におけるソフトウェア内の発見フォールト数 $\{N(t), t \geq 0\}$ は以下の常微分方程式によって記述されるものと仮定する．

$$\frac{dN(t)}{dt} = b(t)\{a - N(t)\}. \tag{1}$$

ここで，$b(t)(>0)$ は時刻 t におけるフォールト発見率を，a はオープンソースソリューションに潜在する総フォールト数を表す．

一般的に，オープンソースソリューションのテスト工程の初期段階においては，各オープンソースコンポーネント間の結合状態や整合性が不安定であり，オープンソースソリューション全体としての機能が十分に発揮できない状況が想定される．このように，大規模オープンソースソリューションの特徴を考えた場合，そのフォールト発見事象は，テスト工程の初期段階において不規則な状態となり，時間の経過とともに安定していくと考えられる．こうした不規則性を，標準化された Gauss 型白色雑音によって近似的に表現する．また，Wiener 過程の分散の性質から，時間の経過とともに Gauss 型白色雑音が増大していく傾向があが，本論文では，オープンソースソリューションの特徴を考慮するために，Gauss 型白色雑音の変化量を表す $\mu(t)$ を導入する．

上記のことから，フォールト発見率 $b(t)$ に不規則性を導入すると，式 (1) は，

$$\frac{dN(t)}{dt} = \{b(t) + \sigma\gamma(t)\mu(t)\}\{a - N(t)\}, \tag{2}$$

となる．ここで，σ (>0) は定数パラメータであり，$\gamma(t)$ は解過程の Markov 性を保証するために標準化された Gauss 型白色雑音である．さらに，$\mu(t)$ はオープンソースソリューション全体の安定性を示す成長関数を表す．式 (2) を，以下の Itô 型の確率微分方程式 [8, 9] に拡張して考える．

$$dN(t) = \{b(t) + \frac{1}{2}\sigma^2\mu(t)^2\}\{a - N(t)\}dt + \sigma\mu(t)\{a - N(t)\}dW(t). \tag{3}$$

式 (3) の確率微分方程式を初期条件 $N(0)=0$ の下で Itô の公式を用いて変換すると，

$$N(t) = a\left\{1 - \exp\left(-\int_0^t b(s)ds - \sigma\mu(t)W(t)\right)\right\}, \tag{4}$$

となる．

本論文では，フォールト発見率 $b(t) = b_1(t)$, $b(t) = b_2(t)$, およびオープンソースソリューション全体の安定性を示す成長関数 $\mu(t)$ は，次式を満たすものとする．

$$\int_0^t b_1(s)ds = (1 - \exp[-bt]), \tag{5}$$

$$\int_0^t b_2(s)ds = (1 - (1+bt)\exp[-bt]), \tag{6}$$

$$\mu(t) = \exp[-ct]. \tag{7}$$

ここで，bはソフトウェアフォールト1個あたりのフォールト発見率を，cは安定性に関する成長係数を表す．

式(4)で定義された$W(t)$の性質は，次の通りである．

(1) $W(t)$はGauss過程である．

(2) $W(t)$の平均および分散は，それぞれ

$$\mathrm{E}[W(t)] = 0, \mathrm{Var}[W(t)] = \sigma^2 t, \tag{8}$$

により与えられる．

(3) $W(t)$は定常独立増分をもつ．

(4) Pr[$W(0)$=0]=1.

3 信頼性評価尺度

3.1 発見フォールト数の期待値と分散

任意の時刻tにおける発見フォールト数の期待値$\mathrm{E}[N(t)]$および分散$\mathrm{Var}[N(t)]$は，ソフトウェア信頼性を評価する上で重要な尺度となる．これらは，Wiener過程$W(t)$の密度関数が，

$$f(W(t)) = \frac{1}{\sqrt{2\pi t}} \exp\left\{-\frac{W(t)^2}{2t}\right\}, \tag{9}$$

であることから，任意の時刻tにおける発見フォールト数の期待値$\mathrm{E}[N(t)]$は，

$$\mathrm{E}[N(t)] = a\left\{1 - \exp\left(-\int_0^t b(s)ds + \frac{\sigma^2\mu(t)^2}{2}t\right)\right\}, \tag{10}$$

となる．

式(10)から，

$$\lim_{t\to\infty} \mathrm{E}[N(t)] = a, \tag{11}$$

となり，時刻tを∞と考えた場合の発見フォールト数はaとなることが分かる．ここでは，オープンソースソリューションは，複数の特定バージョンのOSSから構成されるものと仮定する．

同様にして，$N(t)$の分散も次のように求めることができる[10].

$$\mathrm{Var}[N(t)] = \mathrm{E}[\{N(t) - \mathrm{E}[N(t)]\}^2]. \tag{12}$$

3.2 不完全デバッグ率

不完全デバッグ率は，オープンソースソリューションの安定性に関する時間変化を把握するのに有益な尺度である．また，不完全デバッグ率が小さな値を取ることは，それだけオープンソースソリューションが安定しており，信頼度が向上したと判断できることになる．テスト時刻tにおける不完全デバッグ率は，以下のように導出できる．

$$\Pr[N(t+\Delta t) \le x | N(t) = x] = \Phi\left[\left\{\log\frac{\int_0^t b(s)ds}{\int_0^{t+\Delta t} b(s)ds}\right\} \Big/ \sigma\mu(t+\Delta t)\sqrt{\Delta t}\right]. \tag{13}$$

ここで，xは時刻tにおける累積フォールト発見数であり，微小時間区間$\Delta t(t>0)$において，xを超過しない確率として定義する．また，$\Phi(\cdot)$は標準正規分布関数を表し，

$$\Phi(x) = \frac{1}{2\pi}\int_\infty^x \exp\left(-\frac{y^2}{2}\right)dy, \tag{14}$$

で定義されるものである．

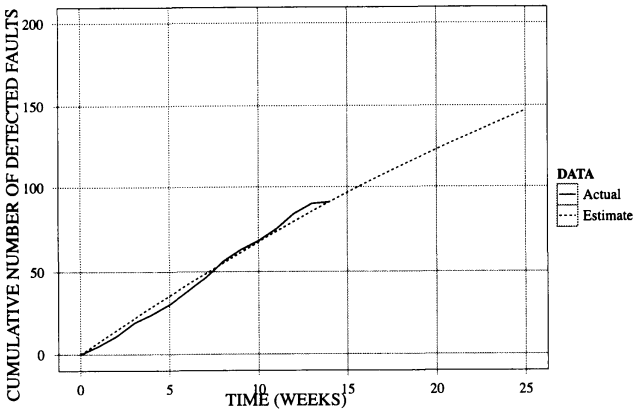

図 1： 推定された累積フォールト発見数の期待値，$\hat{\mathrm{E}}[N(t)]$, $b(t) = b_1(t)$.

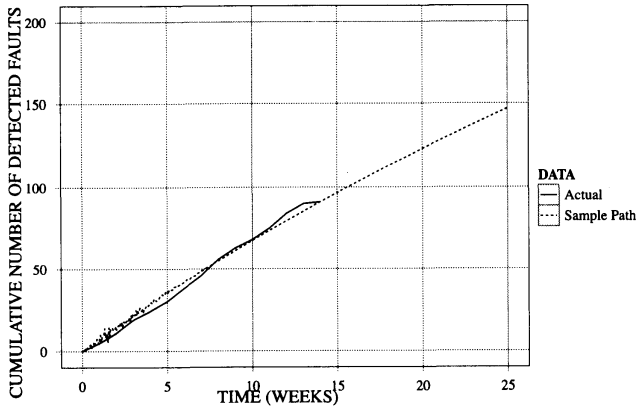

図 2： 推定された累積フォールト発見数のサンプルパス，$\widehat{N}(t)$, $b(t) = b_1(t)$.

4 数値例

本論文では，Apache HTTP Server[11]，Apache Tomcat[12]，MySQL[13] を組み合わせ，JSP（JavaServer Pages）技術を利用した大規模オープンソースソリューションを構築する場合を考える．一例として，大規模オープンソースソリューションのテスト工程を想定するために，実際の Apache HTTP Server，Apache Tomcat，MySQL のオープンソースプロジェクトにおけるバグトラッキングシステム上に登録されたフォールトデータを適用した数値例を示す．

4.1 信頼性評価結果

まず，$b(t) = b_1(t)$ の場合における累積フォールト発見数の期待値の推定値 $\hat{\mathrm{E}}[N(t)]$ を図 1 に示す．また，このときの推定された累積フォールト発見数のサンプルパス $\widehat{N}(t)$ を図 2 に示す．図 2 から，テスト工程の初期においては，フォールト発見現象の不規則性が大きく，時間の経過とともに安定する様子が確認できる．同様に，$b(t) = b_2(t)$ の場合における累積フォールト発見数の期待値の推定値 $\hat{\mathrm{E}}[N(t)]$ を図 3 に示す．また，このときの推定された累積フォールト発見数のサンプルパス $\widehat{N}(t)$ を図 4 に示す．図 4 から，S 字関数の特徴により，フォールト発見現象の不規則性の影響が長く続き，時間の経過とともに緩やかに安定する様子が確認できる．

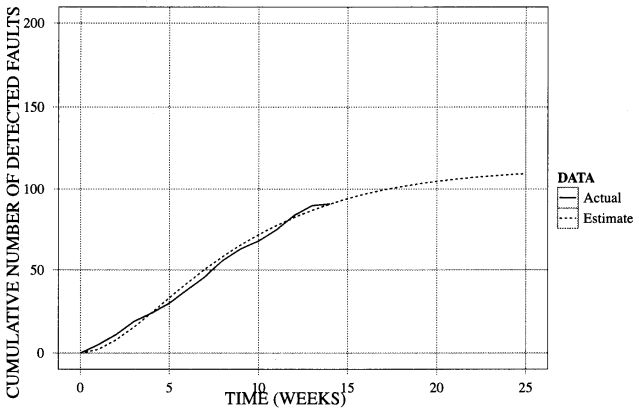

図 3： 推定された累積フォールト発見数の期待値，$\hat{\mathrm{E}}[N(t)]$, $b(t) = b_2(t)$.

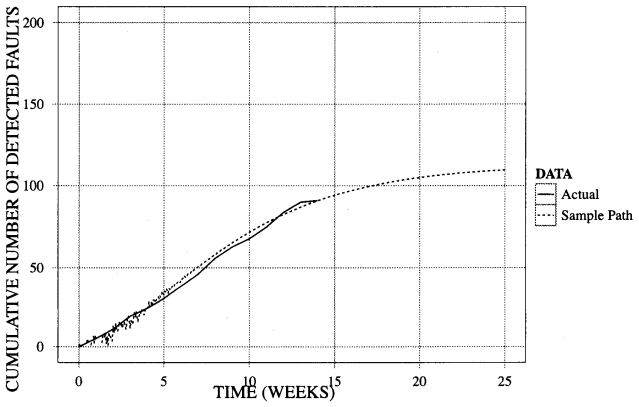

図 4： 推定された累積フォールト発見数のサンプルパス，$\widehat{N}(t)$, $b(t) = b_2(t)$.

4.2 不完全デバッグ率に対するモデルパラメータの感度分析，$b(t) = b_1(t)$

不完全デバッグ率に対する提案モデルに含まれるパラメータに対する感度分析結果を示す．まず，$b(t) = b_1(t)$ の場合におけるモデルに含まれるパラメータ a, c, および σ を固定した状態で，パラメータ b を変化させた場合における感度分析結果とともに，モデルに含まれるパラメータ a, b, および σ を固定した状態で，パラメータ c を変化させた場合における感度分析結果を図 5 に示す．

図 5 から，本モデルに含まれるパラメータ b および c の組み合わせにより，広い範囲の不完全デバッグ率を描くことが可能であることが確認できる．すなわち，パラメータ b の値が大きくなれば不完全デバッグ率が早期に収束し，オープンソースソリューションが安定することが分かる．また，パラメータ c の値が大きい場合も，不完全デバッグ率が早期に収束し，オープンソースソリューションが安定する様子が確認できる．

4.3 不完全デバッグ率に対するモデルパラメータの感度分析，$b(t) = b_2(t)$

同様に，$b(t) = b_2(t)$ の場合において，モデルに含まれるパラメータ a, c, および σ を固定した状態で，パラメータ b を変化させた場合における感度分析結果とともに，モデルに含まれるパラメータ a, b, および σ を固定した状態で，パラメータ c を変化させた場合における感度分析結果を図 6 に示す．

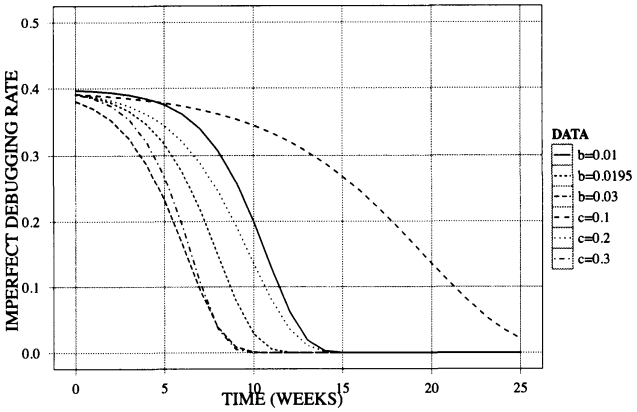

図 5： パラメータ b および c に対する感度分析結果，$b(t) = b_1(t)$.

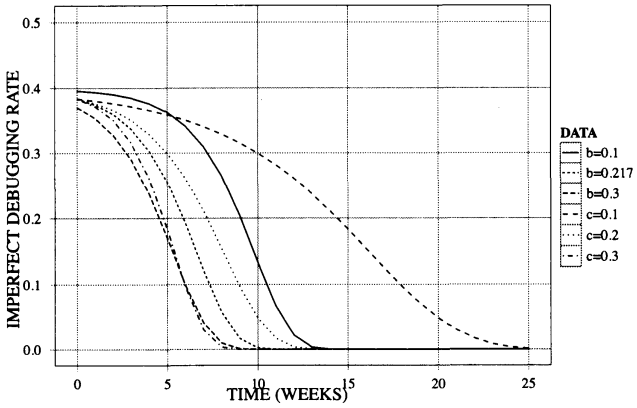

図 6： パラメータ b および c に対する感度分析結果，$b(t) = b_2(t)$.

図 5 および図 6 を比較した場合において，S 字曲線の特徴から，$b(t) = b_1(t)$ よりも $b(t) = b_2(t)$ の方が比較的緩やかに変化している様子が確認できる．

オープンソースソリューションの OSS コンポーネントを結合する段階における不規則性を評価することはソフトウェア開発管理者にとって重要であり，これらの感度分析結果から，不完全デバッグ率を利用することにより，オープンソースソリューション全体の安定性を把握する尺度として利用できるものと考える．

5 むすび

本論文では，大規模オープンソースソリューションに対する確率微分方程式モデルについて議論した．特に，オープンソースソリューションのテスト工程の初期段階においては，各オープンソースコンポーネント間の結合状態や整合性が不安定であり，オープンソースソリューション全体としての機能が十分に発揮できない状況が想定される．こうした大規模オープンソースソリューションの特徴を包括するために，オープンソースソリューション全体の安定性を示す成長パラメータを導入し，確率微分方程式に基づいた SRGM を構築した．また，実際の OSS のバグトラッキングシステムに登録されているフォールト発見数データに対する数値例を示した．さらに，提案された確率微分方程式モデルから新たな信頼性評価尺度として不完全デバッグ率を導出し，モデルに含まれるパ

ラメータに対する感度分析結果を示した．

近年，コスト削減，短納期といった観点から複数のOSSを組み合わせて，一つの大規模オープンソースソリューションを開発する事例が増加している．本論文における提案モデルは，こうしたOSSを利用したオープンソースソリューションの信頼性評価法として利用できるものと考える．特に，不完全デバッグ率を利用することにより，オープンソースソリューション全体の安定性を評価することが可能となる．

OSSが急速に普及し始めている現在，OSSの信頼性に関する指標を提示することが重要であると考える．これまでは，複数のOSSから構成されたオープンソースソリューションの信頼性を動的かつ定量的に評価する試みが行われていなかったため，本論文で提案した信頼性評価手法を適用することにより，より高品質なオープンソースソリューションの開発に結びつくものと考える．

謝辞

本研究の一部は，文部科学省科学研究費基盤研究 (C)（課題番号 22510150）の援助を受けたことを付記する．

参考文献

[1] E-Soft Inc., Internet Research Reports. [Online]. Available: http://www.securityspace.com/s_survey/data/index.html

[2] ソフトウェア情報センター研究会報告書, オープンソースソフトウエアの利用状況調査／導入検討ガイドラインの公表について, 東京, 2004.

[3] S. Yamada, *Software Reliability Models: Fundamentals and Applications* (in Japanese), JUSE Press, Tokyo, 1994.

[4] A. MacCormack, J. Rusnak, and C.Y. Baldwin, "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code," *Informs Journal of Management Science*, vol. 52, no. 7, pp. 1015–1030, 2006.

[5] G. Kuk, "Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List," *Informs Journal of Management Science*, vol. 52, no. 7, pp. 1031–1042, 2006.

[6] Y. Zhoum, J. Davis, "Open source software reliability model: an empirical approach," *Proceedings of the workshop on Open Source Software Engineering (WOSSE)*, vol. 30, no. 4, 2005, pp. 67–72.

[7] P. Li, M. Shaw, J. Herbsleb, B. Ray and P. Santhanam, "Empirical Evaluation of Defect Projection Models for Widely-deployed Production Software Systems," *Proceedings of the 12th International Symposium on the Foundations of Software Engineering (FSE-12)*, 2004, pp. 263–272.

[8] L. Arnold, *Stochastic Differential Equations–Theory and Applications*, John Wiley & Sons, New York, 1974.

[9] E. Wong, *Stochastic Processes in Information and Systems*, McGraw–Hill, New York, 1971.

[10] S. Yamada, M. Kimura, H. Tanaka, and S. Osaki, "Software reliability measurement and assessment with stochastic differential equations," *IEICE Trans. Fundamentals*, vol. E77-A, no. 1, pp. 109-116, Jan. 1994.

[11] The Apache HTTP Server Project, The Apache Software Foundation. [Online]. Available: http://httpd.apache.org/

[12] Apache Tomcat, The Apache Software Foundation. [Online]. Available: http://tomcat.apache.org/

[13] MySQL, Oracle Corporation and/or its affiliates. [Online]. Available: http://www.mysql.com/